Expressions regulars

Les expressions regulars (sovint anomenades RegExp o RegEx) són una seqüència de caràcters que **defineixen un patró de cerca.** Aquest patró es pot utilitzar en operacions del tipus cercar i reemplaçar, oferint inclús capacitat de captura de grups per realitzar uns reemplaçaments molt potents.

Exemple:
Patró: ç
Expressió regular /ç/i (on i és un argument per indicar que és un case-insensitive)

Usos expressions regulars

Exemple d'usos de les expressions regulars. A banda de validar dades, a les expressions regulars se'ls poden donar molts altres usos, per exemple, els següents:

- **Normalitzar textos:** eliminar espais en blanc duplicats, salts de línia erronis (la següent línia comença en minúscula), etc.
- **Canviar els formats:** substituir cometes normals per tipogràfiques, que requereixen conèixer on comença i on acaba l'element.
- **Convertir un text d'un tipus en un altre:** extreure la informació d'una consulta SQL i convertir-la en una cadena de text en format JSON.
- **Realitzar cerques de fitxers amb patrons concrets** (utilitats del sistema operatiu Unix/Linux)

- **Creació de parsers per extreure dades de fitxers amb formats coneguts de text pla** (XML, JSON...) i convertir-les en objectes per poder manipular-los.
- Creació d'intèrprets de nous llenguatges de programació.
- Creació de plantilles on se substituiran valors clau pels valors de les variables que ens interessin.

Però nosaltres ara mateix el farem servir com a validació de dades utilitzant

Expressions regulars Javascript

El patró d e cerca es pot utilitzar per a les operacions de cerca de text i substitució de text.
Vegem a continuació alguns mètodes **dels objectes de tipus RegExp** que permeten realitzar diferents operacions amb expressions regulars:

- **exec**: mètode de RegExp que executa una cerca i **retorna un array amb la informació de la cadena trobada** (posició 0) i, si n'hi ha, grups de captura (resta de posicions); si no troba la cadena, retorna null.

```
const pattern = /e/;
const buscador = pattern.exec("The best things in life are free!");
```

o directament

```
/e/.exec("The best things in life are free!");
```

- **test:** mètode de RegExp que cerca una coincidència a la cadena i **retorna true si la troba o false** en cas contrari.

```
const pattern = /e/;
let buscador= pattern.test("The best things in life are free!");
```

o directament

```
/e/.test("The best things in life are free!");
```

Ara, mètodes **dels objectes de tipus String** més importants:

- **search**: mètode de String que cerca a la cadena de text el patró passat com argument. En cas de trobar-se alguna coincidència **retorna la posició** (dintre de la cadena) on comença la coincidència o el valor -1 si no s'ha trobat. En cas de trobar-se múltiples coincidències només es té en compte la primera d'aquestes i la resta són ignorades.

```
let text= "caçadors de paraules caçadors de paraules";
let n = text.search(/ç/i);
```

- **replace**: mètode de String que executa la cerca i la **reemplaça per la cadena de reemplaçament especificada.**

```
let text= "caçadors de paraules caçadors de paraules";
```

```
let n = text.replace(/ç/i, "ss" );
```

Modificadors

Modifiers can be used to perform case-insensitive more global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Exemple:
let result = text.match(/is/g);

Regular Expression Patterns

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

Exemple de buscar a b o c
let result = text.match(/[abc]/g);

Metacharacters

Metacharacter	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Exemple: 1,0,0
let text = "Give 100%!";
let result = text.match(/\d/g);

Per fer proves:

https://www.w3schools.com/js/js_regexp.asp

Quantifiers define quantities

Quantifiers define quantities:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*

Exemple: troba 1,10,10
let text = "1, 100 or 1000?";
let result = text.match(/10?/g);

JavaScript RegExp Object

L'objecte RegExp es fa servir per fer coincidir text amb un patró.

Consultar

Modifiers, Brackets, Metacharacters, Quantifiers, RegExp Object Properties, RegExp Object Methods
https://www.w3schools.com/jsref/jsref_obj_regexp.asp

Exemples:

www.codepen.io/ioc-daw-m06/pen/jWjmYL
www.codepen.io/ioc-daw-m06/pen/RrzgYL
www.codepen.io/ioc-daw-m06/pen/YwoQKq

Referencies

xuleta expressions regulars
https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_Expressions
https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/RegExp